John von Neumann Institute for Computing

High Performance FEM Simulation via FEAST and Application to Parallel CFD via FEATFLOW

Christian Becker, Sven H. M. Buijssen, Susanne Kilian, Stefan Turek

published in

NIC Symposium 2001, Proceedings,
Horst Rollnik, Dietrich Wolf (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **9**, ISBN 3-00-009055-X, pp. 493-502, 2002.

http://www.fz-juelich.de/nic-series/volume9

High Performance FEM Simulation via FEAST and Application to Parallel CFD via FEATFLOW

Christian Becker, Sven H. M. Buijssen, Susanne Kilian, and Stefan Turek

Institut for Applied Mathematics
University of Dortmund, 44227 Dortmund, Germany
E-mail: featflow@featflow.de

The aim of this project is the development of efficient and robust numerical methods for the simulation of PDEs, particularly for incompressible flows in industrial applications. Since corresponding configurations lead to huge systems of nonlinear equations with fully nonstationary behaviour, the CPU requirements are so high such that special data and matrix structures and hardware-oriented implementation techniques have to be realized. The resulting FEAST software is able to exploit a significant percentage of the potentially available computing power of more than 1 Gflop/s (per single processor) in combination with powerful FEM discretization and parallel multigrid solution techniques.

1 Description of the Project

The aim of this project is to develop efficient numerical methods and implementation techniques for the simulation of PDEs with special emphasis on complex flows which are described by the incompressible Navier-Stokes equations. These components have to be integrated into a CFD software package for industrial applications which is capable of predicting the flow in complex situations. Preliminary results of industrial partners based on simulations with different CFD codes and our own experience show that existing research codes and particularly commercial codes have severe problems to provide sufficiently reliable predictions for important flow quantities in the **fully nonstationary** case. Even on today's supercomputers[5], the required CPU time is often too large while the results are still too inaccurate.

In Figure 1, the results are shown for a typical steady 3D turbulence calculation with the RNG-k-ϵ model which costs 6,5 h CPU time on a SGI Origin2000 with 6 processors

Quantity	Exp.	Sim.	Error
Drag ('c_d')	0.165	0.317	**92 %**
Lift ('c_l')	-0.083	-0.127	**53 %**

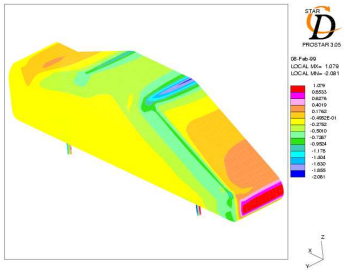

Figure 1. Experiment vs. CFD simulation by DaimlerChrysler obtained with commercial software

for only about 500.000 grid cells. The significant differences between the results from simulation and experiment indicate that much finer computational meshes are needed which in turn will lead to even larger CPU time. Since these codes usually do not contain optimal multigrid solvers, the ratio between number of grid points and required CPU time increases more than linearly, such that 10 times more unknowns are expected to require at least 20 - 30 times more CPU time. The shown geometrical configuration is still quite simple and nonsteady flow behaviour is not considered at all, so it can be imagined that the simulation of realistic time-dependent flow behaviour is impossible for most of the existing CFD tools, even on recent supercomputers.

Based on the research codes FEAST[1] and FEATFLOW[10], a new simulation tool for industrial flow configurations is being created which includes modern numerical and algorithmic components for adaptive control of the discretization in space and time, special FEM discretizations and highly efficient multigrid as well as nonlinear iteration techniques.[8] Besides these mathematical tasks we additionally have to apply improved software technologies and implementation strategies which are adapted to modern processor design.

2 Computational Bottlenecks and the FEAST Software

One of the main components in iterative solvers are matrix-vector (MV) applications. Hereby, *sparse* concepts are the standard techniques in FEM codes (and others): Depending on the programming language, the matrix entries plus index informations are stored as long arrays, containing the 'nonzero entries' only. For an overview of such techniques, see for instance SPARSKIT[11] and the literature cited therein. While this *sparse* approach can be applied for general meshes and arbitrary numberings of unknowns, no explicit advantage of (possible) highly structured parts can be gained. Consequently, a massive loss of performance compared to the possible peak rates may occur since – at least for large problems with more than 100,000 unknowns – no 'caching in' and 'pipelining' can be exploited. In this case, the higher cost of memory access will dominate the resulting MFlop/s rates.

To demonstrate this failure, we show some examples for the FEATFLOW code which is one of the most efficient simulation tools for incompressible flow on general domains.[5] We apply FEATFLOW to a car configuration (ASMO2D) and measure the MFlop/s rates for the MV multiplication inside of the multigrid solver for the momentum equation.[8] The results in Table 1 show that the computational performance is weak and depends strongly on the problem size and the kind of numbering – **T**wo **L**evel ordering, **C**uthill-**M**cKee, **STO**chastic - while the number of arithmetic operations and memory accesses are identical.

The FEAST software is conceptualized to combine such highly tuned linear algebra tools with sophisticated FEM simulation strategies.[9] In contrast to many other approaches which often aim to develop flexible software for research or educational purposes, the FEAST software is designed for high performance applications with industrial background, particularly for *Computational Fluid Dynamics* (CFD). Consequently, our emphasis lies more on 'efficiency' and 'robustness' and less on aspects like 'easy implementable', 'flexible' or 'most modern programming language'. One of the most important principles in FEAST is the consequent application of *(recursive) Divide and Conquer* strategies: The

computer	#unknowns	**TL**	**CM**	**STO**
	13,688	20	22	**19**
Sun E450	54,256	15	17	**13**
($\geq$ 250 MFlop/s)	216,032	14	16	**6**
	862,144	15	16	**4**

Table 1. MFlop/s rates of *sparse* MV multiplication in FEATFLOW for different numberings

solution of a 'global' problem is recursively split into smaller independent subproblems on *patches* as part of the complete set of unknowns. While on certain 'anisotropic' parts the usual *sparse* techniques may be applied (for instance, if local adaptivity is employed), we try to exploit the much higher performance possible on the other, highly structured patches. Consequently, the intention is to minimize the number of 'sparse areas' and to apply preferably most numerical linear algebra tasks on such 'structured patches' via the SPARSEBANDEDBLAS techniques. Hence, the major tasks for realizing such a simulation tool are:

1. Design of the 'skeleton' for the recursive splitting into local/global levels.
2. Implementation of the typical FEM facilities on the 'low level' patches.
3. Development of 'reference element solvers' on the 'low level' patches.

In view of their typically excellent convergence behaviour, multigrid methods seem to be most suited for the solution of many PDEs. However, as the previous examples have shown, multigrid on general domains has often poor computational efficiency, at least if the implementation is based on the standard *sparse* techniques. As a result from our performance measurements[6], the realistic MFlop/s rates are often in the range of 1 MFlop/s only, even on modern high performance workstations. Moreover, the linear relationship between problem size and CPU time is hardly realizable, due to the problem-size dependent performance rates of the *sparse* components. Additionally, the robust treatment of complex mesh structures with locally varying details is often hard to achieve by typical 'Black Box' components.

Concerning parallelization some further serious problems occur: The parallel efficiency is often 'bad' and the achieved performance is far below peak, since the solution of the coarse grid problem leads to a large communication overhead. Furthermore, the relation between 'local' arithmetic operations and 'global' data transfer is poor, in general. Besides these computational aspects, the parallelization of the global smoothers – SOR, ILU – cannot be done efficiently because of their inherent recursive character. So, smoothing can only be performed blockwise, which may lead to a deterioration of the convergence rates. Further, the behaviour of such blockwise smoothing is hard to predict for complicated geometries with local and/or global anisotropies. Motivated by these facts, we developed the more general strategy SCARC for solving discretized PDEs aiming for the following goals:

The parallel efficiency shall be high due to a non-overlapping decomposition and a low communication overhead. The convergence rates are required to be independent of the mesh size h, the complexity of the domain and the number of subdomains N, and they shall be in the range of typical multigrid convergence rates (as $\rho_{MG} \sim 0.1$). Further,

the method shall be easily implementable and uses only existing standard methods. The approach shall accomplish the treatment of complicated geometries with local anisotropies (large aspect ratios) without impairment of the overall convergence rates.

The underlying idea is to '*hide recursively all anisotropies in single subdomains*' combined with '*block Jacobi/Gauß-Seidel smoothing*' within standard multigrid. This approach is based on the numerical experience that these 'simple' block-oriented schemes perform well as soon as all occurring anisotropies are locally hidden, i.e., if the local problems on each block are solved (more or less) exactly. This procedure ensures the global robustness. On the other hand, this also means that the local solution quality in each block can significantly improve the global convergence behaviour. These ideas are combined with hierarchical data and matrix structures, which exploit tensorproduct-like meshes on each *macro* to achieve high performance rates for the linear algebra components in the local (multigrid) solvers. Consequently, all solution processes are recursively organized via sequences of more 'local' steps until the lowest level is reached, for instance a single *macro* with the described generalized tensorproduct mesh. The complete SCARC approach[13] can be characterized as follows:

1. **Sca**lable (with respect to 'quality and number of local solution steps at each stage')
2. **R**ecursive ('independently' for each stage in the hierarchy of partitioning)
3. **C**lustering (for building blocks via 'fixed or adaptive blocking strategies')

3 Numerical Results

We perform some test calculations for the Pressure-Poisson problem. Figure 2 shows a typical macro decomposition (= coarse grid) for the 2D configuration NCC1701-D and a zoom of some macro element (= 1 quadrilateral) directly at the boundary of the vessel. During the mesh generation process, we have applied locally anisotropic refinement in normal direction towards the boundary such that boundary layers can be better resolved while at the same time the tensorproduct topology is locally preserved.

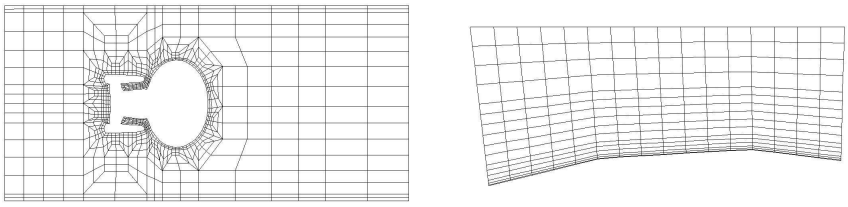

Figure 2. Macro decomposition and anisotropically refined macro (3 refinement levels) for the NCC1701-D configuration

Due to the local grid distortion, special multigrid components have to be employed which work efficient and robust w.r.t. such mesh anisotropies. In the framework of

SCARC as generalized multigrid approach, particularly on parallel platforms, the corresponding local problems (on each refined macro of the coarse grid) are treated with special linewise Gauß-Seidel schemes as smoothing operations inside of an optimized local multigrid solver. Table 2 shows typical (parallel) multigrid rates for the solution of Pressure-Poisson problems within the framework of an nonstationary Navier-Stokes approach via discrete projection methods.[8] The global multigrid convergence rates are more or less independent of the degree of (local) mesh distortion while the resulting CPU times, i.e. the MFlop/s rates, are not yet completely optimized.

$\#NEQ$	Dirichlet 'Velocity'		Neumann 'Pressure'	
	$AR \approx 10$	$AR \approx 10^6$	$AR \approx 10$	$AR \approx 10^6$
$210,944$	0.17 (**8**)	0.18 (**8**)	0.21 (**9**)	0.15 (**8**)
$843,776$	0.17 (**8**)	0.17 (**8**)	0.20 (**9**)	0.17 (**8**)
$3,375,104$	0.18 (**9**)	0.19 (**9**)	0.22 (**10**)	0.22 (**10**)
$13,500,416$	0.19 (**9**)	0.18 (**9**)	0.23 (**10**)	0.23 (**10**)

#P	CPU(s)
4	158.2
8	65.13
16	36.32
32	26.79
64	17.97

Table 2. Global multigrid rates for different aspect ratios and boundary conditions, computing time for a problem with 843,776 unknowns on the Cray T3E with different numbers of CPUs)

The multigrid convergence rates in Table 2 show the robustness and efficiency of this parallel SCARC approach. The resulting CPU times in the right most table already indicate that high efficiency rates will be achieved, which will be further improved by optimizing the administration of the local multigrid solver and particularly by including the corresponding SPARSEBANDEDBLAS techniques: At the moment, the MFlop/s rates for the local multigrid solvers on each tensorproduct mesh are of order 25 MFlop/s and shall be improved by the optimized SPARSEBANDEDBLAS up to 50–100 MFlop/s per processor.

While the numerical components seem to be sufficiently efficient, the computational efficiency in terms of the MFlop/s rates still requires further improvement. Table 3 shows the recent efficiency rates for components from the optimized SPARSEBANDEDBLAS library; here, we can additionally distinguish between the cases of general and of constant matrix entries. It is remarkable that special algorithms and implementation techniques had to be developed for the TGS preconditioner on the CrayT90. It turns out that multigrid, even with this very robust smoother, can be very efficiently performed. Additionally, these results show that the actual MFlop/s rates can significantly depend on the local problem size due to different cache exploitation. This fact can be employed when applying the global SCARC solver since the choice of the local number of macros and the corresponding problem size can be adapted to these effects.

4 Parallel Version of the Existing FEATFLOW Solver

In the meantime, our work on a parallel version of FEATFLOW for CFD problems has been completed. The numerical methods applied have been described by Turek[8] and already been implemented sequentially within the FEATFLOW package.

Before giving some examples of problems with industrial background this implementation has recently been applied to, let us first briefly describe the overall mathematical

2D case	N	DAXPY-I	SBB-V	SBB-C	**MG-V**	**MG-C**
DEC 21264	65^2	205 (178)	538	795	370	452
(667 MHz)	257^2	224 (110)	358	1010	314	487
'**ES40**'	1025^2	**78 (11)**	**158**	**813**	**185**	**401**
Hitachi	65^2	173 (82)	238	391	191	266
(375 MHz)	257^2	143 (29)	243	388	198	260
'**SR8000**'	1025^2	**144 (7)**	**226**	**390**	**200**	**267**
AMD K7	65^2	203 (195)	101	556	122	355
(850 MHz)	257^2	29 (27)	78	241	72	166
'**ATHLON**'	1025^2	**31 (10)**	**64**	**236**	**58**	**126**
Cray T3E	65^2	156 (41)	82	392	85	282
(666 MHz)	257^2	47 (16)	73	294	70	172
'**SciLib**'	1025^2	**30 (7)**	**47**	**283**	**52**	**142**
Cray T90	65^2	627 (500)	855	415	183	163
(440 MHz)	257^2	698 (544)	933	887	276	258
'**SDTSOL**'	1025^2	**699 (553)**	**939**	**1092**	**423**	**419**

Table 3. MFlop/s rates for different problem sizes (#NEQ = number of grid points) for sparse indexed DAXPY (linear access, distributed access in parenthesis), matrix-vector applications (SBB), and complete multigrid solver (TGS smoother); additionally, we can distinguish between variable (V) and constant (C) matrix entries (Poisson problems on orthogonal meshes); special libraries are used on the Cray machines.

background. The underlying model are the instationary incompressible 3D Navier-Stokes equations. They are discretized by a finite element approach ($\tilde{Q}_1/Q_0$ ansatz[4]). Stabilization of the convection term is done by applying an upwind scheme. For time discretization the Fractional-step-θ scheme is used. In order to realize adaptive time stepping, its solutions are compared with those from a Crank-Nicolson scheme. The nonlinear subproblems in each time step are split by applying a discrete projection method (which can be interpreted as a discrete analogue of schemes proposed by Chorin and Van Kan). Using a fixed point defect correction method the resulting nonlinear Burgers equations are linearized and finally solved by a parallel multigrid method. The remaining discrete Pressure-Poisson equations are solved using a parallel conjugate gradient scheme which is preconditioned by multigrid.

In order to parallelize the multigrid method the mesh is split into parallel blocks by a graph-oriented partitioning tool (see example in figure 8). Consistency with the sequential algorithm (MV application, restriction, prolongation) is guaranteed through local communication between at most two parallel blocks (this is possible because of the face oriented $\tilde{Q}_1/Q_0$ ansatz). As mentioned earlier, the inherent recursive character of global smoothers prevents a direct parallelization. Therefore, the global smoothing is replaced by smoothing within each parallel block only (block ILU smoother). To minimize the communication overhead solving the coarse grid problem, it is treated on a single processor with an optimized sequential algorithm. The cost are two global communications (setting up right side vector and propagation of solution vector).

4.1 An Industrial Problem

The program has been applied to a bunch of problems, one of them arising from steel industry. The behaviour of molten steel within a mould was supposed to be simulated. A

mould is a component of a slab continuous casting production set (Figure 3). It is one of the critical sections during production of slabs. Molten steel pours into the mould through a small nozzle in the upper part (Figure 5), solidifies due to cooling at the surrounding walls and leaves the mould at its bottom. The final aim is to simulate the complete production process including cooling down and electro-magnetic stirring. For now, these additional technical measures are neglected and we concentrate only on the simulation of the complex flow patterns within the mould.

Figure 4 shows the problem's coarse grid. It only contains moderate aspect ratios ($AR = 30$). After four refinements we got a problem size of 29,706,240 unknowns in space. It took 2900 time steps to reach the endpoint $T = 36s$. Using 256 processors on the Cray T3E-1200 this simulation was performed within 20 hours. With 1 pre- and post-smoothing step for the linearized Burgers equations the average multigrid convergence rate was 0.03. For the discrete Pressure-Poisson equations the multigrid preconditioner performed 4 pre- and post-smoothing steps, leading to typical convergence rates for the preconditioned conjugate gradient method of 0.31.

Some snapshots of the resulting nonstationary non-periodic flow patterns can be seen in Figure 6 (isosurfaces of vertical velocity) and Figure 7 (particle tracing).[2]

4.2 Deterioration of Solver Due to Block Smoothers

As explained, there is a price to pay for the 'easy' way of parallelizing the smoothing process by replacing global smoothing with block smoothers. A deterioration of convergence rates can be expected, but it is hard to predict the degree of deterioration for complicated geometries with local and/or global anisotropies. Therefore, a complex problem derived from automotive aerodynamics was studied: flow around a 3D sample car (denoted as ASMO3D). Because the industrial partner was/is interested in predictions on lift and drag coefficients on the car's surface, a body-fitted mesh had to be used. As a result, the solvers had to cope with large areas of distorted hexahedrons and locally high aspect ratios.

The coarse grid consisted of 840 elements and was refined three times such that a

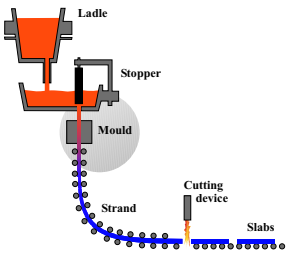

Figure 3. Scheme slab continuous casting production set

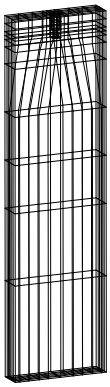

Figure 4. Coarse grid used. Four times uniformly refined it consists of 2,949,120 elements.

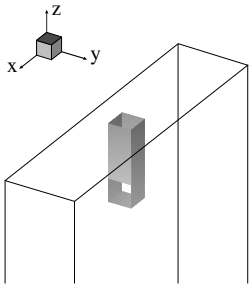

Figure 5. Magnified view of the upper domain part with inflow nozzle

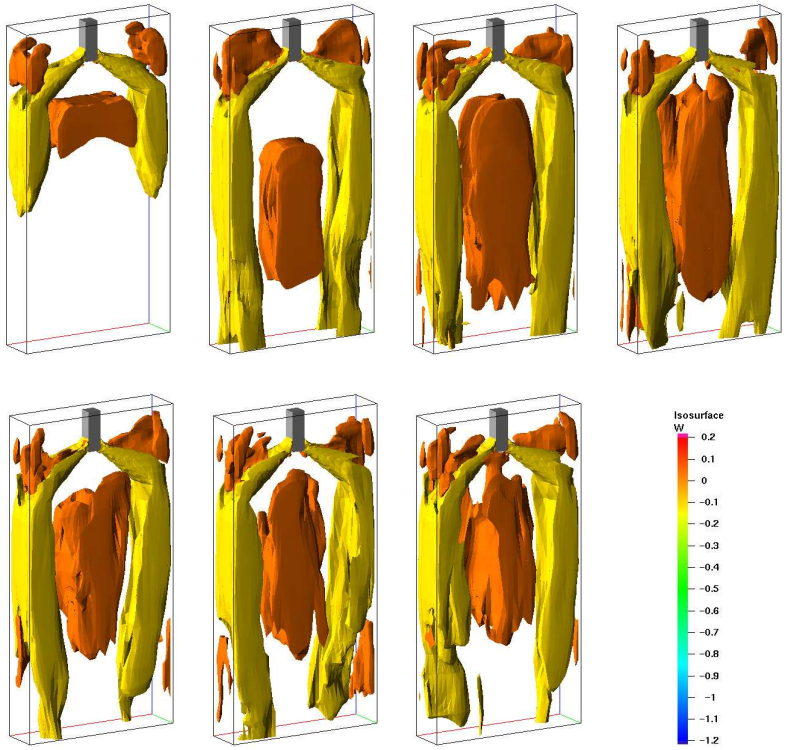

Figure 6. Flow pattern of molten steel within a mould. Visualization using isosurfaces of vertical velocity ($\Delta t = 5,5s$)

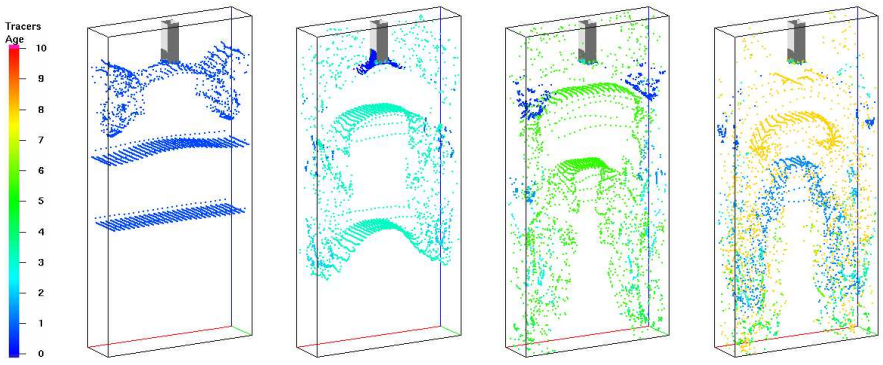

Figure 7. Flow pattern of molten steel within a mould. Particle tracing visualization ($\Delta t = 3,5s$)

problem with 4,368,576 unknowns in space was gained. A time period $T = [0, 0.5]$ was simulated with different numbers of processors ranging from 1 to 64. For sample grid partitions into 8 and 64 parallel blocks respectively see Figure 8. Solving linearized Burgers equations required 1 pre- and post-smoothing step, discrete Pressure-Poisson equation as much as 8.

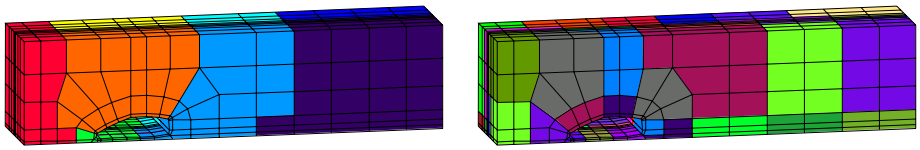

Figure 8. Sample grid partitions for ASMO3D problem into 8 and 64 parallel blocks respectively

These scaling tests were performed on several architectures: Alphacluster ALiCE (Wuppertal), Cray T3E-1200, Linuxcluster (Heidelberg), Sun Enterprise 3500 (Dortmund). Unfortunately, the run time scaling for this problem (see Figure 9) is rather disappointing. Increasing the number of processors from 4 to 64 on the Cray T3E-1200 gives only a speedup of approximately 5. Scaling on the remaining architectures was even worse.

It turned out that this is not mainly caused by an increasing communication overhead, but by the fact that the average number of iterations solving the discrete Pressure-Poisson equation increases from 2.2 (1 processor) to 6.2 (64 processors). Trying other solving methods (classical multigrid, cg with additive instead of multiplicative multigrid preconditioning) and smoothers (Jacobi, SOR) gave even worse results. Thus, already at moderate aspect ratios the solution process suffers significantly from the deterioration of smoothing property due to the use of 'simple' block smoothers.

5 Outlook

The central point of this project is the development of mathematical components – FEM discretizations, adaptivity and multigrid solvers – and their realization in a software pack-

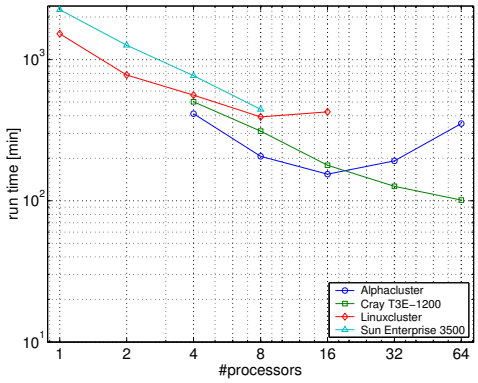

Figure 9. Run time for ASMO3D using different numbers of processors on different platforms

age which directly includes tools for parallelism and hardware-adapted high-performance in low level kernel routines. The code generation uses the new FEAST software in order to achieve highest computational efficiency. These software developments can be viewed as 'basic research' in the field of mathematical software for PDEs. It is the special goal in this project to realize and to optimize the SPARSEBANDEDBLAS concepts for specific parallel computers (CrayT3E, CrayT90, Sun HPC machines, Linux cluster) and to adapt the mathematical components to complex configurations. Since the corresponding geometries may lead to severe mesh distortions, for instance near boundaries, special multigrid components for highly-stretched body fitted meshes have been developed which allow high numerical efficiency and robustness in the multigrid-like SCARC approach.

At the moment, we are completing the 2D Navier-Stokes solver which is based on conforming bilinear FEM and discrete projection methods. This code will be integrated into FEAST such that the parallel/sequential high-performance tools of the SPARSEBANDEDBLAS will be directly available. Besides that, we will continue our work with the parallel 3D adaptation of the FEATFLOW solver which is presently applied on several parallel computers to prototypical configurations similar to the shown geometries. This parallel 3D code is our candidate for all further developments which aim to incorporate the high-performance FEAST techniques into this CFD tool in order to achieve highest computational efficiency on modern computers in combination with the 'best' numerical approaches.

References

1. Becker, Chr.: FEAST - *The Realization of Finite Element Software for High-performance Applications*, PhD Thesis, to be published.
2. BUIJSSEN, S.: *Numerische Analyse eines parallelen 3-D-Navier-Stokes-Lösers*. Diplomarbeit, Universität Heidelberg, 2001.
3. Kilian, S., Turek, S.: *An example for parallel* SCARC *and its application to the incompressible Navier-Stokes equations*, Proc. ENUMATH-97, Heidelberg, October 1997, World Science Publ., 1998.
4. RANNACHER, R., TUREK, S.: *Simple Nonconforming Quadrilateral Stokes Element*. Numerical Methods for Partial Differential Equations, 8(2):97–111, March 1992.
5. Schäfer, M., Rannacher, R., Turek, S.: *Evaluation of a CFD benchmark for laminar flows*, Proc. ENUMATH-97, Heidelberg, October 1997, World Science Publ., 1998.
6. Turek, S. et al.: *Performance rating via the* FEAST INDICES, Computing, 63, 283 - 297, 1999.
7. Turek, S. et al.: *Proposal for* SPARSE BANDED BLAS *techniques*, Preprint 99–11, U Heidelberg, SFB 359, 1999, submitted to *Int. J. of High Performance Computing*.
8. Turek, S.: *Efficient solvers for incompressible flow problems: An algorithmic and computational approach*, LNCSE 6, Springer-Verlag, 1999.
9. Turek, S.: *Trends in processor technology and their impact on Numerics for PDE's*, Preprint 99–31, U Heidelberg, SFB 359, 1999, submitted to *Numerische Mathematik*.
10. Turek, S.: **FEATFLOW** . *Finite element software for the incompressible Navier–Stokes equations: User Manual, Release 1.2*, 1999.
11. SPARSKIT (by Y. Saad), `www.cs.umn.edu/Research/arpa/SPARSKIT`.